CCITT SGXV
Working Party XV/1
Experts Group for ATM Video Coding

Document AVC-385
October 1992

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG 92/
October 1992

Title: Core Experiment on Cell Loss resilence by using frequency scanning

Source: O. Poncin (BELGACOM), B. Hammer (SIEMENS), G. Schamel (HHI)
On behalf of the Eureka VADIS project

Purpose: Proposal

Appendix F: Cell Loss Experiments

F.4 Core Experiment on Cell Loss resilience by using frequency scanning

The purpose of this experiment is to compare the impacts of cell losses on both the block scanning (Basic Mode) and the frequency scanning with MUVLC techniques.

The output bitstream of the frequency scanning mode will be split up into 2 layers (with 2 different CLR's). The layer 1 will contain the (n-1) MSB's of the 4X4 LF coefficients; the layer 2 will contain the LSB of the 4X4 LF coefficients and the full amplitude of the remaining 48 coefficients.

No concealment technique will be applied at the decoder side.

F.4.1 Global parameters of the experiment

The core experiment will be carried out by using the following parameters:

chroma_format:	4:2:0
picture_structure:	frame picture
bit_rate:	4 Mbit/s
group of pictures structure:	N=12 (15), M=3
prediction	field/frame adaptive
transform	field/frame adaptive
number of frames	50 (60)

The simulation of Cell Loss will be performed according to Appendix F - Section F.1 of TM2.

The parameters for Cell Loss will be :

-Total bit rate : 4 Mbit/s
-High priority bit rate : 1.6 Mbit/s (40% of total).

F.4.2 Description of MUVLC

The MUVLC is described in Appendix Q - Section Q.8.2. and Appendix I - Section I.8.3.

A C-program of the MUVLC algorithm will be distributed to interested parties by e-mail (send requests to B. Hammer (Siemens), ha@bvax4.zfe.siemens.de, or J. De Lameillieure (HHI), grunaaie@w204zrz.zrz.tu-berlin.de).

F.4.3 Syntax modifications

For purpose of the frequency scanning experiments the slice layer, the macro block layer and the block layer must be changed as indicated below:

F.4.3.1 Slice Layer

```
slice() {
  slice_start_code
  quantizer_scale
          ::::
    no change
          ::::
  extra_bit_slice
  slice_control()
  slice_data()
}
```

F.4.3.2 Slice Control Layer

```
slice_control() {
  for (mb=0; mb<44; mb ++) {
    while(nextbits() == '0000 0001 111')
      macroblock_stuffing
          ::::
          same syntax as in macroblock() of TM2
          ::::
    if (macroblock_pattern)
      coded_block_pattern()
  }
}
```

F.4.3.3 Slice Data Layer

```
slice_data() {
  for (coef_i=1;coef_i<65; coef_i ++) {
    buf_size = 0
    for (mb = 0; mb < 44; mb++) {
      for(block_i=1; block_i<5; block_i++) {
        if (pattern_code[mb][block_i]) {
          c_buf[buf_size] = dct_coef[mb][block_i][coef_i]
            buf_size++
          }
        }
      }
      for (mb = 0; mb < 44; mb++) {
       if (pattern_code[mb][5]) {
         c_buf[buf_size] = dct_coef[mb][5][coef_i]
           buf_size++
         }
       }
       for (mb = 0; mb < 44; mb++) {
        if (pattern_code[mb][6]) {
          c_buf[buf_size] = dct_coef[mb][6][coef_i]
            buf_size++
          }
        }
     if (coef_i<17)
       muvlc0(c_buf,buf_size)
     else
       muvlc1(c_buf,buf_size)
   }
  while(nextbits()!='0000 0000 0000 0000 0000 0001')
   next_start_code()
}
```

```
muvlc0(c_buf,buf_size) {
pc_layer1 = pc_code(c_buf,buf_size)                                    3
  for(class=pc_layer1; class=0; class--) {
      pl_layer1 = pl_code(c_buf, buf_size)                             3
     run = 0
     for (buf_i = 0; buf_i < buf_size; buf_i ++) {
       if(!c_coded[buf_i]) {
         if (((c_buf[buf_i] >> class) & 0x1) == 1) {
           rl = rl_code(run,pl_layer1)
              ncb_layer1 = ncb_code_MSB(c_buf[buf_i],pc_layer1)        1-7
              ncb_layer2 = ncb_code_LSB(c_buf[buf_i],pc_layer1)        1
           c_coded[buf_i] = 1
            run = 0
          }
          else
            run++
         }
     }
     eoc = eoc_code(run,pl_layer1)
   }
}
```

where ncb_code_LSB is the LSB of the coefficient;
ncb_code_MSB are the coefficient bits without the LSB.

```
muvlc1(c_buf,buf_size) {
pc_layer2 = pc_code(c_buf,buf_size)                                    3
  for(class=pc_layer2; class=0; class--) {
      pl_layer2 = pl_code(c_buf, buf_size)                             3
    run = 0
    for (buf_i = 0; buf_i < buf_size; buf_i ++) {
      if(!c_coded[buf_i]) {
        if (((c_buf[buf_i] >> class) & 0x1) == 1) {
          rl = rl_code(run,pl_layer2)
             ncb_layer2 = ncb_code(c_buf[buf_i],pc_layer2)               1-8
          c_coded[buf_i] = 1
           run = 0
         }
         else
           run++
        }
    }
    eoc = eoc_code(run,pl_layer2)
  }
}
```

```
rl_code(run,pl) {
  while (run >= 1<<pl) {
    max_run                                          1        "0"
    run -= 1<<pl
  }
   prefix                                            1        "1"
   run                                               1-8
}
```

```
eoc_code(run,pl) {
  while (run > 0) {
    max_run                                          1        "0"
    run -= 1<<pl
  }
```